Employee Management System (Group 1)

Unit Testing & Whitebox Testing Report

The tech stack of our system is Node.js for the backend, Flutter for the front end, and MongoDB for the database.

Backend

The backend systems were tested using both static code analysis and unit testing.

Static code analysis: **ESLint**

ESLint was used for static code analysis of our Node.js code.
This testing extension helps us avoid some granular mistakes like unnecessary try-catch blocks, unused variables, and libraries that were used but not imported, and, most importantly, enforces naming conventions.
Initially, we did not use this and ran into many bugs caused due to above-mentioned problems. ESLint efficiently pointed out the bugs that could have caused the system to run down.

Unit testing: **Mocha.js & Chai**

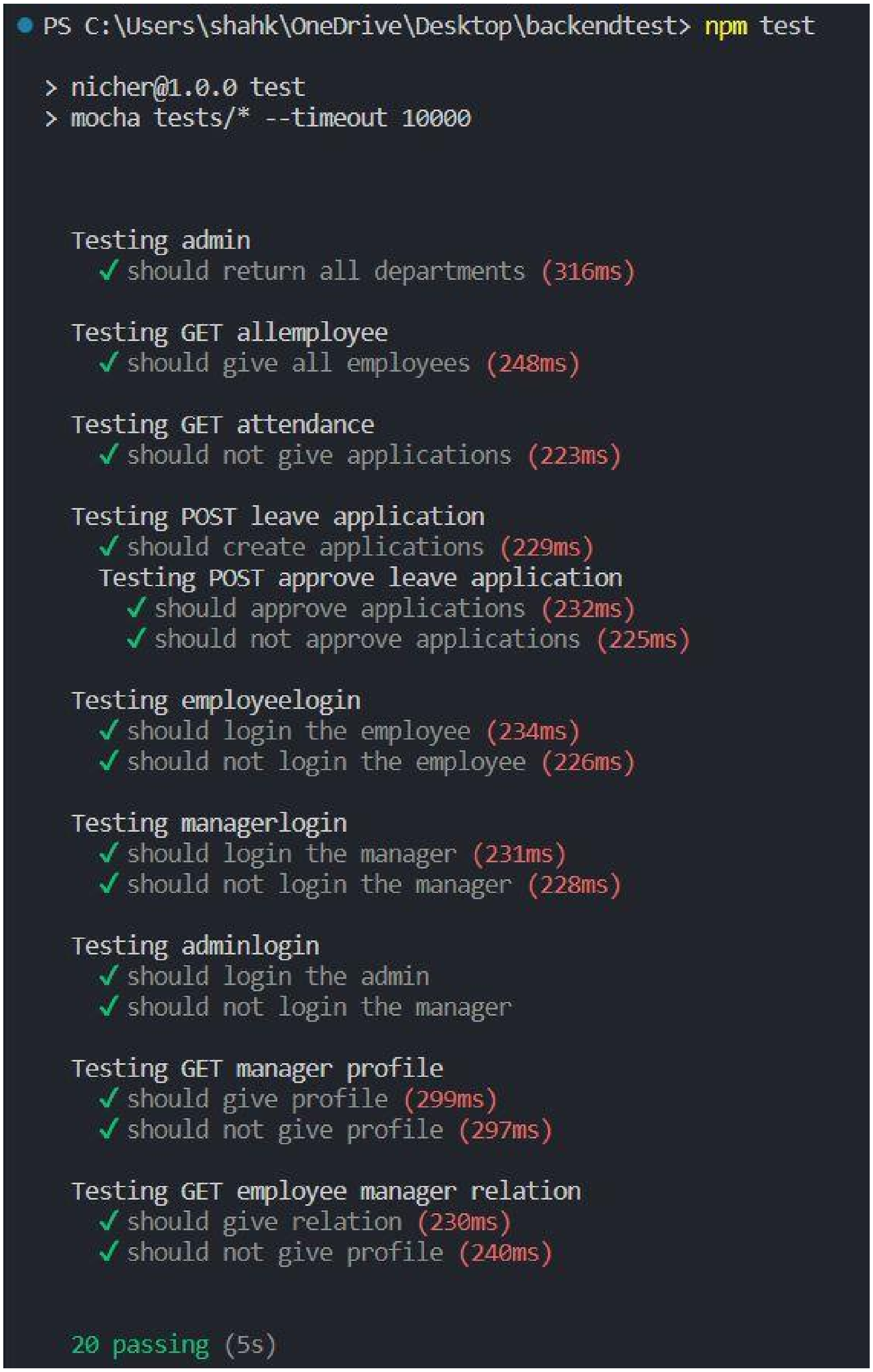
Mocha.js and **Chai** libraries were used for unit testing of javascript code. They provided an efficient way to test all the API services and routes created. Many potential errors were pointed out in this testing, like the database returning NULL values, incorrect datatype considered, and others.
Those errors were solved promptly, and this helped our code become robust.

Example test cases used in Unit testing:

The following list is not exhaustive. It is just the sample of test cases used.

```
let chai = require('chai');
let chaiHttp = require('chai-http');

chai.should();
chai.use(chaiHttp);

const host = 'http://localhost:8000';
const path = '/employee'

describe('Testing GET allemployee', ()=>{
    it('should give all employees', (done)=>{           // correct username and password
        chai
        .request(host)
        .get(path+'/allemployee')
        .end((err,res)=>{
            res.should.have.status(200);
            res.body.should.be.a('array');
            done();
            });
    });
})


describe('Testing GET attendance', ()=>{
    it('should give data', (done)=>{           // correct id
        chai
        .request(host)
        .get(path+'/attendance/1')
        .end((err,res)=>{
            res.should.have.status(200);
            res.body.should.be.a('object');
            done();
            });
    });
    it('should not give data', (done)=>{            // correct id
        chai
        .request(host)
        .get(path+'/attendance/100')
        .end((err,res)=>{
            res.should.not.have.status(200);
            res.body.should.be.a('object');
            done();
            });
    });
})


describe('Testing POST create new employee', ()=>{
    it('should create new employee', (done)=>{
        chai
        .request(host)
        .post(path)
        .send({
            "EmployeeID": 3,
            "EmployeeName": "Test Employee",
// need to write a unique username everytime
            "UserName": "user",
            "Password": "user",
            "Email": "user",
            "PhoneNo":"1221332",
            "Post":"QA",
            "Salary": 10000,
            "Manager": 1
        })
        .end((err,res)=>{
            res.should.have.status(200);
            res.body.should.be.a('object');
            done();
            });
    });
    it('should not create new employee', (done)=>{            // correct
username and password
        chai
        .request(host)
        .post(path)
        .send({
            "EmployeeID": 3,
            "EmployeeName": "Test Employee",
 // need to write a unique username everytime
            "UserName": "user",
            "Password": "user",
            "Email": "user",
            "PhoneNo":"1221332",
            "Post":"QA",
            "Salary": 10000,
            "Manager": 1
        })
        .end((err,res)=>{
            res.should.not.have.status(200);
            res.body.should.be.a('object');
            done();
            });
    });
})
```

Sample output:

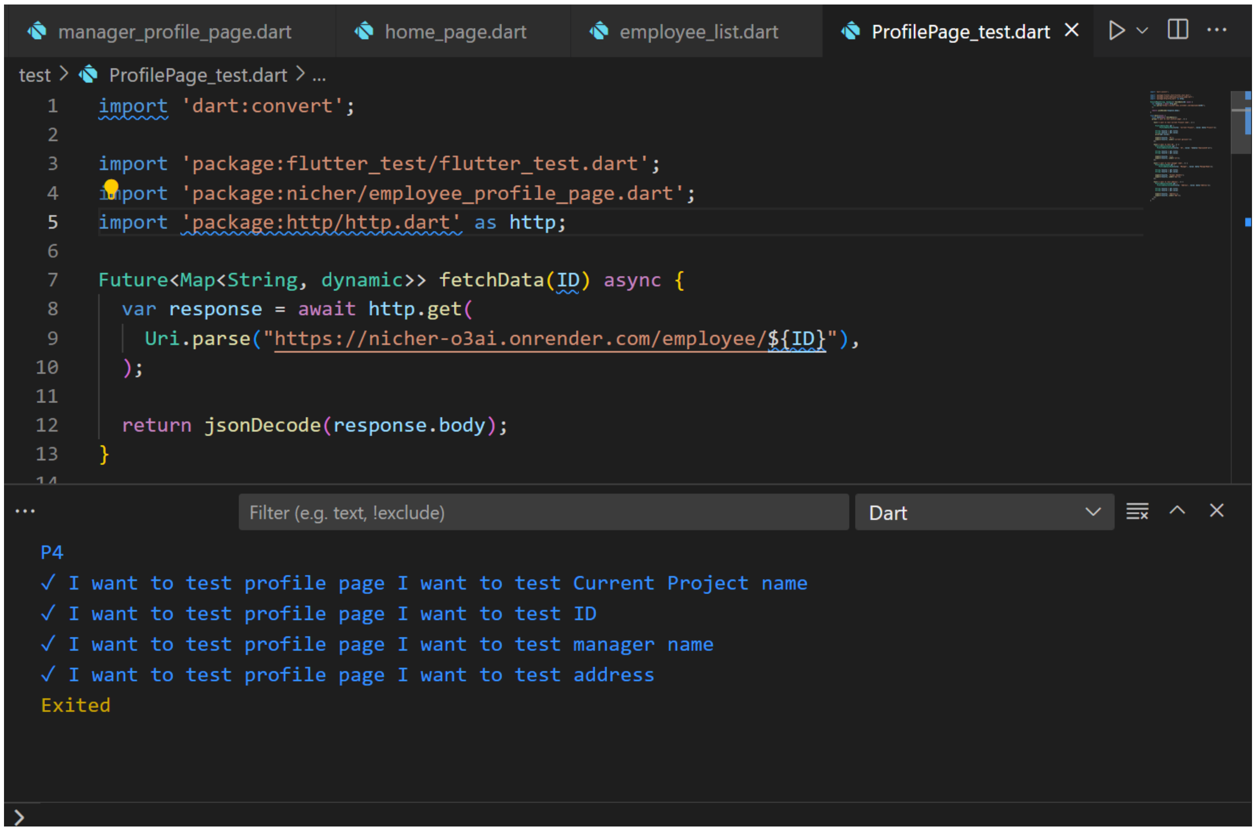


```
● PS C:\Users\shahk\OneDrive\Desktop\backendtest> npm test

  > nicher@1.0.0 test
  > mocha tests/* --timeout 10000



    Testing admin
      ✓ should return all departments (316ms)

    Testing GET allemployee
      ✓ should give all employees (248ms)

    Testing GET attendance
      ✓ should not give applications (223ms)

    Testing POST leave application
      ✓ should create applications (229ms)
      Testing POST approve leave application
        ✓ should approve applications (232ms)
        ✓ should not approve applications (225ms)

    Testing employeelogin
      ✓ should login the employee (234ms)
      ✓ should not login the employee (226ms)

    Testing managerlogin
      ✓ should login the manager (231ms)
      ✓ should not login the manager (228ms)

    Testing adminlogin
      ✓ should login the admin
      ✓ should not login the manager

    Testing GET manager profile
      ✓ should give profile (299ms)
      ✓ should not give profile (297ms)

    Testing GET employee manager relation
      ✓ should give relation (230ms)
      ✓ should not give profile (240ms)


    20 passing (5s)
```

Frontend Testing :

Profile Page

```
import 'dart:convert';

import 'package:flutter_test/flutter_test.dart';
import 'package:nicher/employee_profile_page.dart';
import 'package:http/http.dart' as http;

Future<Map<String, dynamic>> fetchData(ID) async {
  var response = await http.get(
    Uri.parse("https://nicher-o3ai.onrender.com/employee/${ID}"),
  );

  return jsonDecode(response.body);
}

void main()async {
  final data=await fetchData(1);
  group('I want to test profile page', () {

    test('I want to test Current Project name', () {

      ProfileDetailRow pdr =
           ProfileDetailRow(title: 'Current Project', value:
data['Project']);

      String result1 = pdr.title;
      String result2 = pdr.value;
      print(pdr.value);

      expect(result2, 'P4');
      expect(result1, isNot('current pproject'));
    });
    test('I want to test ID', () {
      ProfileDetailColumn pdr =
        ProfileDetailColumn(title: 'ID', value: "${data['EmployeeID']}");

      String result1 = pdr.title;
      String result2 = pdr.value;

      expect(result2, '1');
      expect(result1, isNot('id'));
    });

    test('I want to test manager name', () {
      ProfileDetailRowD pdr =
          ProfileDetailRowD(title: 'Manager', value: data['ManagerName']);

      String result1 = pdr.title;
      String result2 = pdr.value;

      expect(result2, 'Vivaan Chandra');
      expect(result1, isNot('man'));
    });

    test('I want to test address', () {
      ProfileDetailColumnD pdr =
        ProfileDetailColumnD(title: 'Address', value: data['Address']);

      String result1 = pdr.title;
      String result2 = pdr.value;

      expect(result1, 'Address');
      expect(result2, isNot('ABC'));
    });
  });
}
```

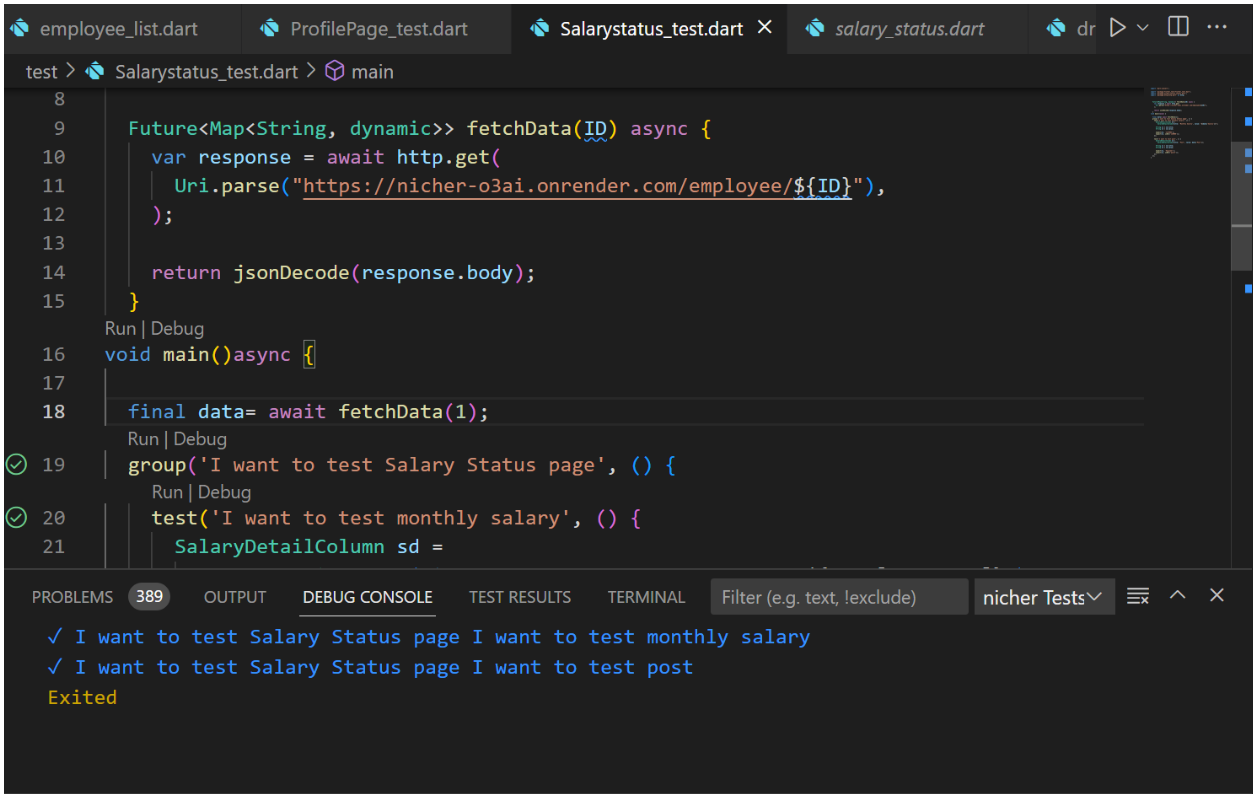

manager_profile_page.dart
home_page.dart
employee_list.dart
ProfilePage_test.dart
test > ProfilePage_test.dart > ...
Filter (e.g. text, !exclude)
Dart

```dart
import 'dart:convert';

import 'package:flutter_test/flutter_test.dart';
import 'package:nicher/employee_profile_page.dart';
import 'package:http/http.dart' as http;

Future<Map<String, dynamic>> fetchData(ID) async {
  var response = await http.get(
    Uri.parse("https://nicher-o3ai.onrender.com/employee/${ID}"),
  );

  return jsonDecode(response.body);
}
```

```
P4
✓ I want to test profile page I want to test Current Project name
✓ I want to test profile page I want to test ID
✓ I want to test profile page I want to test manager name
✓ I want to test profile page I want to test address
Exited
```

Salary Status Page

```dart
import 'dart:convert';

import 'package:flutter_test/flutter_test.dart';
import 'package:nicher/salary_status.dart';
import 'package:http/http.dart' as http;



  Future<Map<String, dynamic>> fetchData(ID) async {
    var response = await http.get(
      Uri.parse("https://nicher-o3ai.onrender.com/employee/${ID}"),
    );

    return jsonDecode(response.body);
  }
void main() async {
  final data= await fetchData(1);
  group('I want to test Salary Status page', () {
    test('I want to test monthly salary', () {
      SalaryDetailColumn sd =
        SalaryDetailColumn(title: 'Monthly Salary', value:
"${data['Salary']}");

      String s1 = sd.title;
      String s2 = sd.value;

      expect(s2, '1174983');
      expect(s2, isNot('120000'));
    });

    test('I want to test post', () {
      SalaryDetailColumn sd =
        SalaryDetailColumn(title: 'Post', value: data['Post']);

      String s1 = sd.title;
      String s2 = sd.value;

      expect(s2, 'Engineer');
      expect(s1, isNot('postt'));
    });
  });
}
```

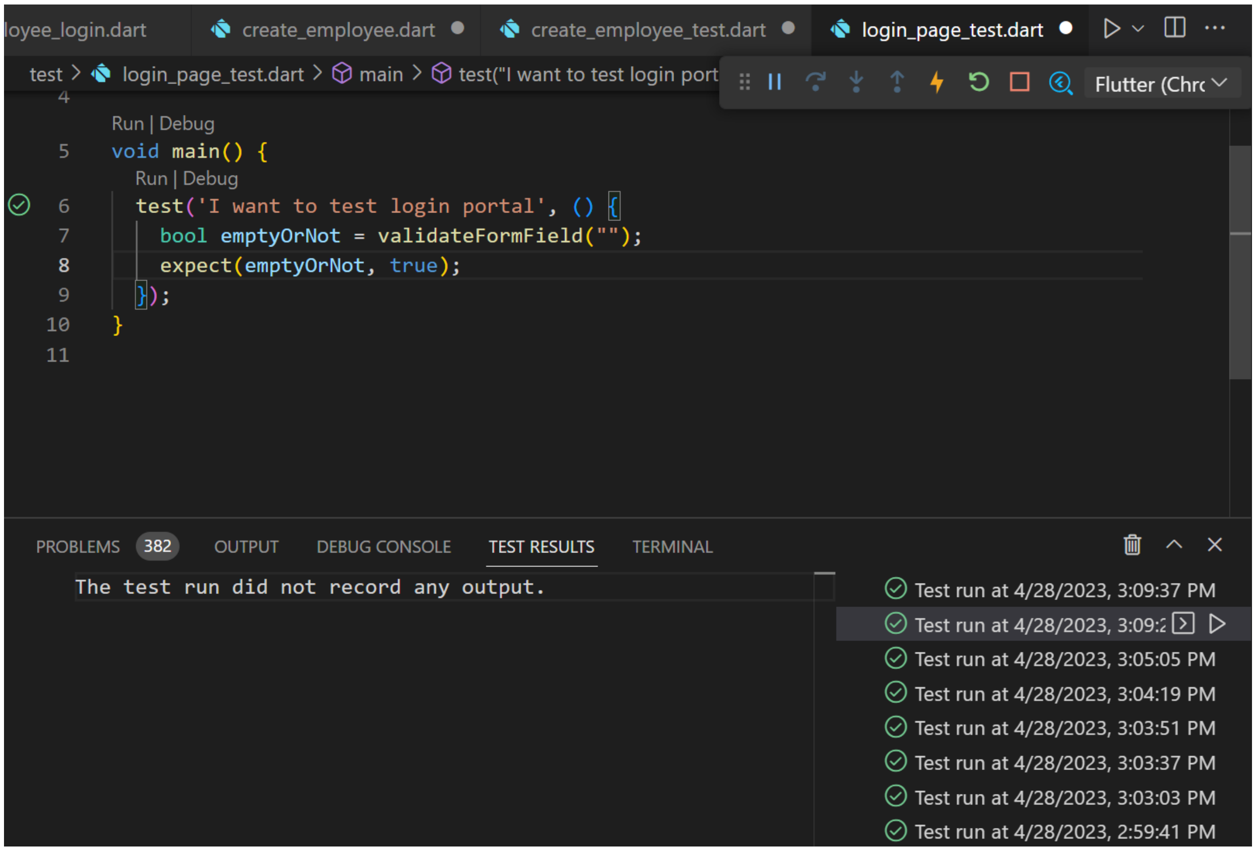


```
  Future<Map<String, dynamic>> fetchData(ID) async {
    var response = await http.get(
      Uri.parse("https://nicher-o3ai.onrender.com/employee/${ID}"),
    );

    return jsonDecode(response.body);
  }
Run | Debug
void main()async {

  final data= await fetchData(1);
  Run | Debug
  group('I want to test Salary Status page', () {
    Run | Debug
    test('I want to test monthly salary', () {
      SalaryDetailColumn sd =
```

```
✓ I want to test Salary Status page I want to test monthly salary
✓ I want to test Salary Status page I want to test post
Exited
```

Login Page

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:nicher/employee_login.dart';

void main() {
  test('I want to test login portal', () {
    bool emptyOrNot = validateFormField("");
    expect(emptyOrNot, true);
  });
}
```



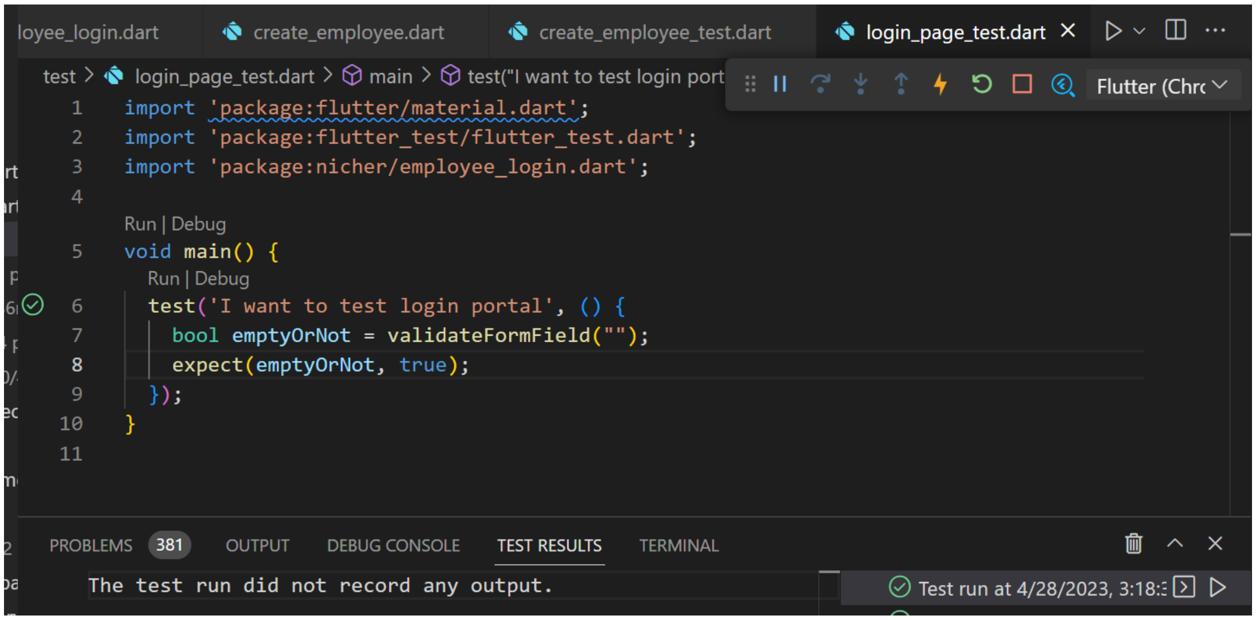
loyee_login.dart
create_employee.dart
create_employee_test.dart
login_page_test.dart
test > login_page_test.dart > main > test("I want to test login port
Flutter (Chrc

```
Run | Debug
void main() {
  Run | Debug
  test('I want to test login portal', () {
    bool emptyOrNot = validateFormField("");
    expect(emptyOrNot, true);
  });
}
```

```
The test run did not record any output.
```

```
Test run at 4/28/2023, 3:09:37 PM
Test run at 4/28/2023, 3:09:2
Test run at 4/28/2023, 3:05:05 PM
Test run at 4/28/2023, 3:04:19 PM
Test run at 4/28/2023, 3:03:51 PM
Test run at 4/28/2023, 3:03:37 PM
Test run at 4/28/2023, 3:03:03 PM
Test run at 4/28/2023, 2:59:41 PM
```

Create employee



loyee_login.dart
create_employee.dart
create_employee_test.dart
login_page_test.dart
test > login_page_test.dart > main > test("I want to test login port
Flutter (Chrc

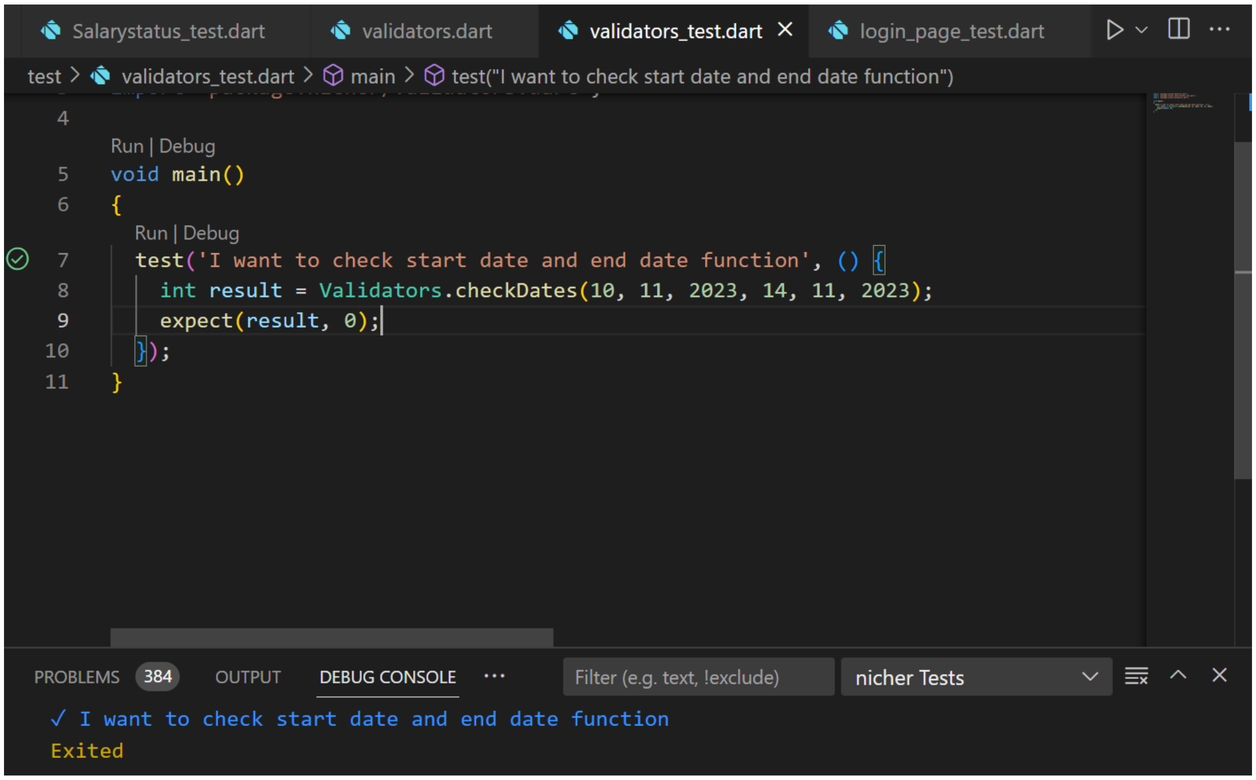
```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:nicher/employee_login.dart';

Run | Debug
void main() {
  Run | Debug
  test('I want to test login portal', () {
    bool emptyOrNot = validateFormField("");
    expect(emptyOrNot, true);
  });
}
```

```
The test run did not record any output.
```

```
Test run at 4/28/2023, 3:18:
```

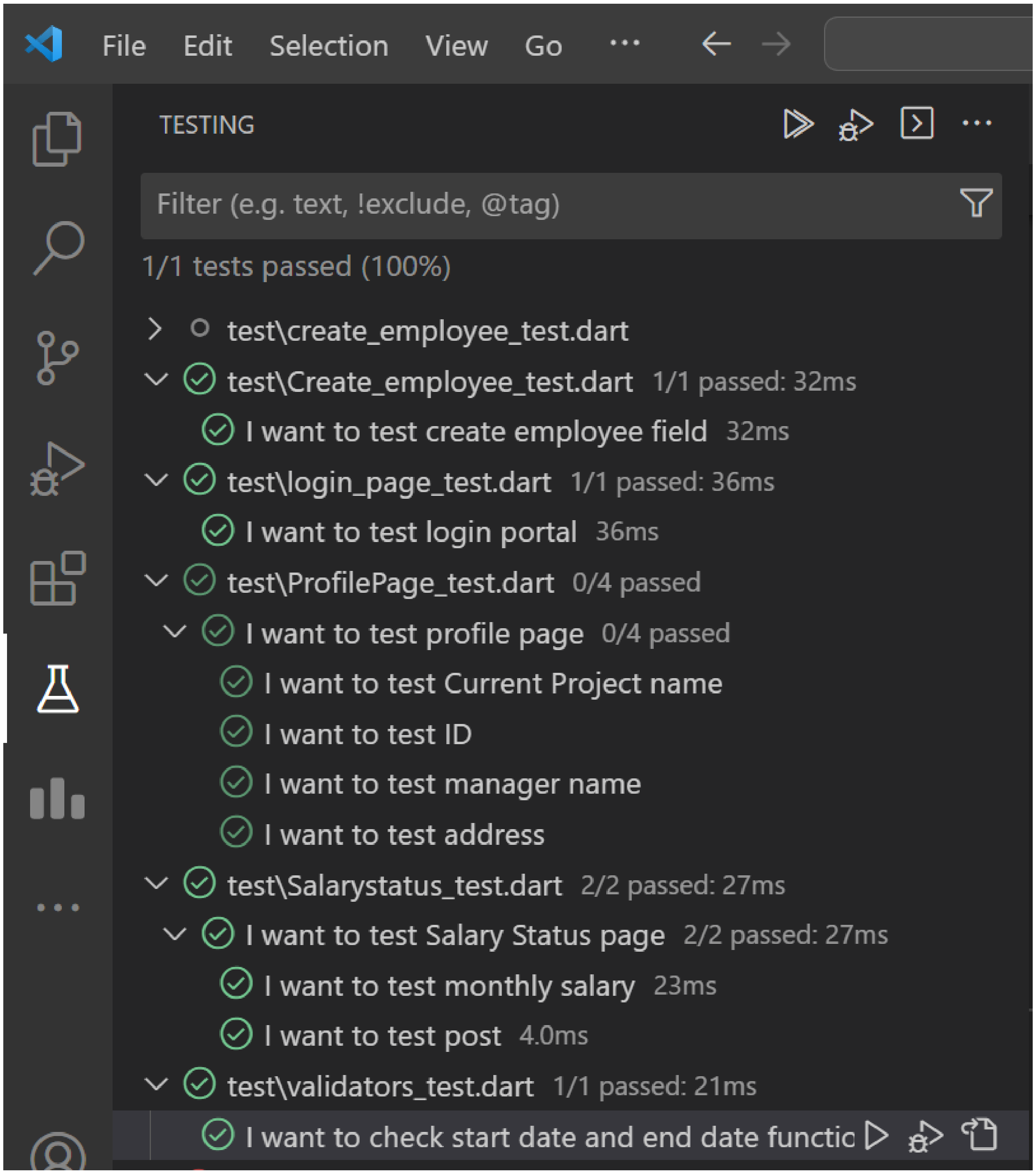
Validators function

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:nicher/validators.dart';

void main()
{
  test('I want to check start date and end date function', () {
    int result = Validators.checkDates(10, 11, 2023, 14, 11, 2023);
    expect(result, 0);
  });
}
```



```
void main()
{
  test('I want to check start date and end date function', () {
    int result = Validators.checkDates(10, 11, 2023, 14, 11, 2023);
    expect(result, 0);
  });
}
```

```
✓ I want to check start date and end date function
Exited
```



1/1 tests passed (100%)

test\create_employee_test.dart
test\Create_employee_test.dart 1/1 passed: 32ms
I want to test create employee field 32ms
test\login_page_test.dart 1/1 passed: 36ms
I want to test login portal 36ms
test\ProfilePage_test.dart 0/4 passed
I want to test profile page 0/4 passed
I want to test Current Project name
I want to test ID
I want to test manager name
I want to test address
test\Salarystatus_test.dart 2/2 passed: 27ms
I want to test Salary Status page 2/2 passed: 27ms
I want to test monthly salary 23ms
I want to test post 4.0ms
test\validators_test.dart 1/1 passed: 21ms
I want to check start date and end date functic